



1  
2  
3  
4

**Document Number: DSP0817**

**Date: 2009-07-14**

**Version: 1.0.0**

5 **IP Interface Profile SM CLP Command Mapping**  
6 **Specification**

7 **Document Type: Specification**  
8 **Document Status: DMTF Standard**  
9 **Document Language: E**

10

11 Copyright notice

12 Copyright © 2006, 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

13 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems  
14 management and interoperability. Members and non-members may reproduce DMTF specifications and  
15 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to  
16 time, the particular version and release date should always be noted.

17 Implementation of certain elements of this standard or proposed standard may be subject to third party  
18 patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations  
19 to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose,  
20 or identify any or all such third party patent right, owners or claimants, nor for any incomplete or  
21 inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to  
22 any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize,  
23 disclose, or identify any such third party patent rights, or for such party's reliance on the standard or  
24 incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any  
25 party implementing such standard, whether such implementation is foreseeable or not, nor to any patent  
26 owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is  
27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  
28 implementing the standard from any and all claims of infringement by a patent owner for such  
29 implementations.

30 For information about patents held by third-parties which have notified the DMTF that, in their opinion,  
31 such patent may relate to or impact implementations of DMTF standards, visit  
32 <http://www.dmtf.org/about/policies/disclosures.php>.

33

34

# CONTENTS

35 Foreword ..... 5

36 Introduction ..... 6

37 1 Scope ..... 7

38 2 Normative References..... 7

39 2.1 Approved References ..... 7

40 2.2 Other References..... 7

41 3 Terms and Definitions..... 7

42 4 Symbols and Abbreviated Terms..... 8

43 5 Recipes..... 9

44 6 Mappings..... 9

45 6.1 CIM\_BindsToLANEndpoint..... 9

46 6.2 CIM\_ElementCapabilities ..... 12

47 6.3 CIM\_ElementSettingData ..... 14

48 6.4 CIM\_EnabledLogicalElementCapabilities..... 24

49 6.5 CIM\_HostedAccessPoint ..... 26

50 6.6 CIM\_HostedService ..... 29

51 6.7 CIM\_IPAssignmentSettingData ..... 31

52 6.8 CIM\_IPConfigurationService ..... 34

53 6.9 CIM\_IPProtocolEndpoint ..... 37

54 6.10 CIM\_OrderedComponent ..... 42

55 6.11 CIM\_RemoteAccessAvailableToElement..... 45

56 6.12 CIM\_RemoteServiceAccessPoint..... 47

57 6.13 CIM\_ServiceAffectsElement ..... 50

58 6.14 CIM\_StaticIPAssignmentSettingData ..... 53

59 ANNEX A (informative) Change Log ..... 57

60

## 61 Tables

62 Table 1 – Command Verb Requirements for CIM\_BindsToLANEndpoint..... 10

63 Table 2 – Command Verb Requirements for CIM\_ElementCapabilities ..... 12

64 Table 3 – Command Verb Requirements for CIM\_ElementSettingData ..... 14

65 Table 4 – Command Verb Requirements for CIM\_EnabledLogicalElementCapabilities..... 24

66 Table 5 – Command Verb Requirements for CIM\_HostedAccessPoint ..... 26

67 Table 6 – Command Verb Requirements for CIM\_HostedService ..... 29

68 Table 7 – Command Verb Requirements for CIM\_IPAssignmentSettingData ..... 31

69 Table 8 – Command Verb Requirements for CIM\_IPConfigurationService ..... 35

70 Table 9 – Command Verb Requirements for CIM\_IPProtocolEndpoint ..... 38

71 Table 10 – Command Verb Requirements for CIM\_OrderedComponent ..... 42

72 Table 11 – Command Verb Requirements for CIM\_RemoteAccessAvailableToElement..... 45

73 Table 12 – Command Verb Requirements for CIM\_RemoteServiceAccessPoint..... 47

74 Table 13 – Command Verb Requirements for CIM\_ServiceAffectsElement ..... 51

75 Table 14 – Command Verb Requirements for CIM\_StaticIPAssignmentSettingData ..... 53

76



78

## Foreword

79 The *IP Interface Profile SM CLP Command Mapping Specification* (DSP0817) was prepared by the  
80 Server Management Working Group.

### 81 **Conventions**

82 The pseudo-code conventions utilized in this document are the Recipe Conventions as defined in SNIA  
83 [SMI-S 1.1.0](#), section 7.6.

### 84 **Acknowledgements**

85 The authors wish to acknowledge the following participants from the DTMF Server Management Working  
86 Group:

- 87 • Aaron Merkin – IBM
- 88 • Jon Hass – Dell
- 89 • Khachatur Papanyan – Dell
- 90 • Enoch Suen – Dell
- 91 • Jeff Hilland – HP
- 92 • Christina Shaw – HP
- 93 • Perry Vincent – Intel
- 94 • John Leung – Intel

95

96

## Introduction

97 This document defines the SM CLP mapping for the CIM elements described in the [IP Interface Profile](#).  
98 The information in this specification, combined with the [SM CLP-to-CIM Common Mapping Specification](#)  
99 [1.0](#), is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and  
100 methods described in the [IP Interface Profile](#) using CIM operations.

101 The target audience for this specification is implementers of the SM CLP support for the [IP Interface](#)  
102 [Profile](#).

# 103 IP Interface Profile SM CLP Command Mapping Specification

## 104 1 Scope

105 This specification contains the requirements for an implementation of the SM CLP to provide access to,  
106 and implement the behaviors of, the [IP Interface Profile](#).

## 107 2 Normative References

108 The following referenced documents are indispensable for the application of this document. For dated  
109 references, only the edition cited applies. For undated references, the latest edition of the referenced  
110 document (including any amendments) applies.

### 111 2.1 Approved References

112 DMTF DSP0216, *SM CLP-to-CIM Common Mapping Specification 1.0*,  
113 [http://www.dmtf.org/standards/published\\_documents/DSP0216\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0216_1.0.pdf)

114 DMTF DSP1036, *IP Interface Profile 1.0*,  
115 [http://www.dmtf.org/standards/published\\_documents/DSP1036\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1036_1.0.pdf)

116 SNIA, *Storage Management Initiative Specification (SMI-S) 1.1.0*,  
117 [http://www.snia.org/tech\\_activities/standards/curr\\_standards/smi](http://www.snia.org/tech_activities/standards/curr_standards/smi)

### 118 2.2 Other References

119 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,  
120 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

## 121 3 Terms and Definitions

122 For the purposes of this document, the following terms and definitions apply.

### 123 3.1

#### 124 **can**

125 used for statements of possibility and capability, whether material, physical, or causal

### 126 3.2

#### 127 **cannot**

128 used for statements of possibility and capability, whether material, physical or causal

### 129 3.3

#### 130 **conditional**

131 indicates requirements to be followed strictly in order to conform to the document when the specified  
132 conditions are met

### 133 3.4

#### 134 **mandatory**

135 indicates requirements to be followed strictly in order to conform to the document and from which no  
136 deviation is permitted

- 137 **3.5**  
138 **may**  
139 indicates a course of action permissible within the limits of the document
- 140 **3.6**  
141 **need not**  
142 indicates a course of action permissible within the limits of the document
- 143 **3.7**  
144 **optional**  
145 indicates a course of action permissible within the limits of the document
- 146 **3.8**  
147 **shall**  
148 indicates requirements to be followed strictly in order to conform to the document and from which no  
149 deviation is permitted
- 150 **3.9**  
151 **shall not**  
152 indicates requirements to be followed strictly in order to conform to the document and from which no  
153 deviation is permitted
- 154 **3.10**  
155 **should**  
156 indicates that among several possibilities, one is recommended as particularly suitable, without  
157 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 158 **3.11**  
159 **should not**  
160 indicates that a certain possibility or course of action is deprecated but not prohibited

## 161 **4 Symbols and Abbreviated Terms**

162 The following symbols and abbreviations are used in this document.

- 163 **4.1**  
164 **CIM**  
165 Common Information Model
- 166 **4.2**  
167 **CLP**  
168 Command Line Protocol
- 169 **4.3**  
170 **DMTF**  
171 Distributed Management Task Force
- 172 **4.4**  
173 **IETF**  
174 Internet Engineering Task Force



- 175 **4.5**  
176 **SM**  
177 Server Management
- 178 **4.6**  
179 **SMI-S**  
180 Storage Management Initiative Specification
- 181 **4.7**  
182 **SNIA**  
183 Storage Networking Industry Association
- 184 **4.8**  
185 **UFsT**  
186 User Friendly selection Tag

## 187 **5 Recipes**

188 The following is a list of the common recipes used by the mappings in this specification. For a definition of  
189 each recipe, see the *SM CLP-to-CIM Common Mapping Specification 1.0* ([DSP0216](#)).

- 190 • smStartRSC()
- 191 • smStopRSC()
- 192 • smResetRSC()
- 193 • smShowInstance()
- 194 • smShowInstances()
- 195 • smSetInstance()
- 196 • smShowAssociationInstances()
- 197 • smShowAssociationInstance()
- 198 • smNewInstance()
- 199 • smAddError()
- 200 • smCommandStatus()

201 This mapping does not define any recipes for local reuse.

## 202 **6 Mappings**

203 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in  
204 the [IP Interface Profile](#). Requirements specified here related to support for a CLP verb for a particular  
205 class are solely within the context of this profile.

### 206 **6.1 CIM\_BindsToLANEndpoint**

207 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

208 Table 1 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
209 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
210 verb and target. Table 1 is for informational purposes only; in case of a conflict between Table 1 and

211 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 212 information in Table 1.

213 **Table 1 – Command Verb Requirements for CIM\_BindsToLANEndpoint**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.1.2.
start	Not supported	
stop	Not supported	

214 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 215 `reset`, `set`, `start`, and `stop`.

### 216 6.1.1 Ordering of Results

217 When results are returned for multiple instances of `CIM_BindsToLANEndpoint`, implementations shall  
 218 utilize the following algorithm to produce the natural (that is, default) ordering:

- 219 • Results for `CIM_BindsToLANEndpoint` are unordered; therefore, no algorithm is defined.

### 220 6.1.2 Show

221 This section describes how to implement the `show` verb when applied to an instance of  
 222 `CIM_BindsToLANEndpoint`. Implementations shall support the use of the `show` verb with  
 223 `CIM_BindsToLANEndpoint`.

224 The `show` command is used to display information about the `CIM_BindsToLANEndpoint` instance or  
 225 instances.

#### 226 6.1.2.1 Show Multiple Instances

227 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
 228 to a `show` command issued against `CIM_BindsToLANEndpoint` where only one reference is specified and  
 229 the reference is to an instance of `CIM_LANEndpoint`.

##### 230 6.1.2.1.1 Command Form

```
231 show <CIM_BindsToLANEndpoint multiple objects>
```

### 232 6.1.3 CIM Requirements

233 See `CIM_BindsToLANEndpoint` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 234 mandatory properties.

### 235 6.1.3.1.1 Behavior Requirements

#### 236 6.1.3.1.1.1 Preconditions

237 \$instance contains the instance of CIM\_LANEndpoint which is referenced by  
238 CIM\_BindsToLANEndpoint.

#### 239 6.1.3.1.1.2 Pseudo Code

```
240 &smShowAssociationInstances ( "CIM_BindsToLANEndpoint", $instance.getObjectPath() );
241 &smEnd;
```

### 242 6.1.3.2 Show a Single Instance – CIM\_IPProtocolEndpoint Reference

243 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
244 a `show` command issued against `CIM_BindsToLANEndpoint` where the reference specified is to an  
245 instance of `CIM_IPProtocolEndpoint`. An instance of `CIM_IPProtocolEndpoint` is referenced by exactly  
246 one instance of `CIM_BindsToLANEndpoint`. Therefore, a single instance will be returned.

#### 247 6.1.3.2.1 Command Form

```
248 show <CIM_BindsToLANEndpoint single object>
```

#### 249 6.1.3.2.2 CIM Requirements

250 See `CIM_BindsToLANEndpoint` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
251 mandatory properties.

### 252 6.1.3.2.3 Behavior Requirements

#### 253 6.1.3.2.3.1 Preconditions

254 \$instance contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
255 `CIM_BindsToLANEndpoint`.

#### 256 6.1.3.2.3.2 Pseudo Code

```
257 &smShowAssociationInstances ( "CIM_BindsToLANEndpoint", $instance.getObjectPath() );
258 &smEnd;
```

### 259 6.1.3.3 Show a Single Instance – Both References

260 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
261 a `show` command issued against `CIM_BindsToLANEndpoint` where both references are specified and  
262 therefore the desired instance is unambiguously identified.

#### 263 6.1.3.3.1 Command Form

```
264 show <CIM_BindsToLANEndpoint single object>
```

#### 265 6.1.3.3.2 CIM Requirements

266 See `CIM_BindsToLANEndpoint` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
267 mandatory properties.

268 **6.1.3.3.3 Behavior Requirements**269 **6.1.3.3.3.1 Preconditions**

270 \$instanceA contains the instance of CIM\_LANEndpoint which is referenced by  
 271 CIM\_BindsToLANEndpoint.

272 \$instanceB contains the instance of CIM\_IPProtocolEndpoint which is referenced by  
 273 CIM\_BindsToLANEndpoint.

274 **6.1.3.3.3.2 Pseudo Code**

```
275 &smShowAssociationInstance ( "CIM_BindsToLANEndpoint", $instanceA.getObjectPath(),
276     $instanceB.getObjectPath() );
277 &smEnd;
```

278 **6.2 CIM\_ElementCapabilities**

279 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

280 Table 2 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
 281 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 282 verb and target. Table 2 is for informational purposes only; in case of a conflict between Table 2 and  
 283 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 284 information in Table 2.

285 **Table 2 – Command Verb Requirements for CIM\_ElementCapabilities**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.2.2.
start	Not supported	
stop	Not supported	

286 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 287 `reset`, `set`, `start`, and `stop`.

288 **6.2.1 Ordering of Results**

289 When results are returned for multiple instances of CIM\_ElementCapabilities, implementations shall  
 290 utilize the following algorithm to produce the natural (that is, default) ordering:

- 291
- Results for CIM\_ElementCapabilities are unordered; therefore, no algorithm is defined.

## 292 **6.2.2 Show**

293 This section describes how to implement the `show` verb when applied to an instance of  
 294 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with  
 295 `CIM_ElementCapabilities`.

296 The `show` command is used to display information about the `CIM_ElementCapabilities` instance or  
 297 instances.

### 298 **6.2.2.1 Show Multiple Instances – CIM\_EnabledLogicalElementCapabilities Reference**

299 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
 300 to a `show` command issued against `CIM_ElementCapabilities` where only one reference is specified and  
 301 the reference is to an instance of `CIM_EnabledLogicalElementCapabilities`.

#### 302 **6.2.2.1.1 Command Form**

```
303 show <CIM_ElementCapabilities multiple objects>
```

#### 304 **6.2.2.1.2 CIM Requirements**

305 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 306 mandatory properties.

#### 307 **6.2.2.1.3 Behavior Requirements**

##### 308 **6.2.2.1.3.1 Preconditions**

309 `$instance` contains the instance of `CIM_EnabledLogicalElementCapabilities` which is referenced by  
 310 `CIM_ElementCapabilities`.

##### 311 **6.2.2.1.3.2 Pseudo Code**

```
312 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
313 &smEnd;
```

### 314 **6.2.2.2 Show a Single Instance – CIM\_IPProtocolEndpoint Reference**

315 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 316 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an  
 317 instance of `CIM_EnabledLogicalElementCapabilities`. An instance of  
 318 `CIM_EnabledLogicalElementCapabilities` is referenced by exactly one instance of  
 319 `CIM_ElementCapabilities`. Therefore, a single instance will be returned.

#### 320 **6.2.2.2.1 Command Form**

```
321 show <CIM_ElementCapabilities single object>
```

#### 322 **6.2.2.2.2 CIM Requirements**

323 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 324 mandatory properties.

#### 325 **6.2.2.2.3 Behavior Requirements**

##### 326 **6.2.2.2.3.1 Preconditions**

327 `$instance` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
 328 `CIM_ElementCapabilities`.

329 **6.2.2.2.3.2 Pseudo Code**

```
330 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );
331 &smEnd;
```

332 **6.2.2.3 Show a Single Instance – Both References**

333 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 334 a `show` command issued against `CIM_ElementCapabilities` where both references are specified and  
 335 therefore the desired instance is unambiguously identified.

336 **6.2.2.3.1 Command Form**

```
337 show <CIM_ElementCapabilities single object>
```

338 **6.2.2.3.2 CIM Requirements**

339 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 340 mandatory properties.

341 **6.2.2.3.3 Behavior Requirements**342 **6.2.2.3.3.1 Preconditions**

343 `$instanceA` contains the instance of `CIM_EnabledLogicalElementCapabilities` which is referenced by  
 344 `CIM_ElementCapabilities`.

345 `$instanceB` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
 346 `CIM_ElementCapabilities`.

347 **6.2.2.3.3.2 Pseudo Code**

```
348 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
349     $instanceB.getObjectPath() );
350 &smEnd;
```

351 **6.3 CIM\_ElementSettingData**

352 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

353 Table 3 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
 354 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 355 verb and target. Table 3 is for informational purposes only; in case of a conflict between Table 3 and  
 356 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 357 information in Table 3.

358 **Table 3 – Command Verb Requirements for `CIM_ElementSettingData`**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.3.2.

Command Verb	Requirement	Comments
show	Shall	See 6.3.3.
start	Not supported	
stop	Not supported	

359 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
360 `reset`, `set`, `start`, and `stop`.

### 361 6.3.1 Ordering of Results

362 When results are returned for multiple instances of `CIM_ElementSettingData`, implementations shall  
363 utilize the following algorithm to produce the natural (that is, default) ordering:

- 364 • Results for `CIM_ElementSettingData` are unordered; therefore, no algorithm is defined.

### 365 6.3.2 Set

366 This section describes how to implement the `set` verb when applied to an instance of  
367 `CIM_ElementSettingData`. Implementations shall support the use of the `set` verb for an instance of  
368 `CIM_ElementSettingData` which references an instance of `CIM_IPAssignmentSettingData`.

#### 369 6.3.2.1 Set IsCurrent Property

370 This command form is for when the `set` verb is used to change the value of the `IsCurrent` property of an  
371 instance of `CIM_ElementSettingData` that associates an instance of `CIM_IPAssignmentSettingData` with  
372 an instance of `CIM_IPProtocolEndpoint`. The only valid input value is 1 (Is Current). The `IsCurrent`  
373 property is not modified directly. The mapping interprets this value as a request to assign the referenced  
374 `CIM_IPAssignmentSettingData` to the referenced `CIM_IPProtocolEndpoint` instance using the  
375 `CIM_IPConfigurationService` associated with the `CIM_IPProtocolEndpoint`.

##### 376 6.3.2.1.1 Command Form

```
377 set [-force] <CIM_ElementSettingData single instance> IsCurrent=<propvalue>
```

##### 378 6.3.2.1.2 CIM Requirements

379 See `CIM_ElementSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the  
380 `CIM_IPConfigurationService.ApplySettingToIPProtocolEndpoint` property.

##### 381 6.3.2.1.3 Behavior Requirements

###### 382 6.3.2.1.3.1 Preconditions

383 `$bForce` is true if the “-force” option was specified.

384 Make sure an instance of `CIM_IPAssignmentSettingData` (and not a subclass) is referenced. If a subclass  
385 (such as `StaticIPAssignmentSettingData`) is referenced instead, it is not a valid target.

###### 386 6.3.2.1.3.2 Pseudo Code

```
387 $instance=<CIM_ElementSettingData single instance>
388 // $bForce is true if force option was specified
389 // Make sure the property value specified in the command is valid
390 // 1 (“Is Current”)
391 if (<propvalue> != 1) {
```

```

392     $OperationError = smNewInstance("CIM_Error");
393     //CIM_ERR_FAILED
394     $OperationError.CIMStatusCode = 1;
395     //Software Error
396     $OperationError.ErrorType = 4;
397     //Unknown
398     $OperationError.PerceivedSeverity = 0;
399     $OperationError.OwningEntity = DMTF:SMCLP;
400     $OperationError.MessageID = 0x0000000E;
401     $OperationError.Message = "The value specified for the {1} property is not valid.";
402     $OperationError.MessageArguments = { "IsCurrent" };
403     &smAddError($job, $OperationError);
404     &smMakeCommandStatus($job);
405     &smEnd;
406 }
407 // Make sure an instance of CIM_IPAssignmentSettingData (and not a sub-class)
408 // is referenced, if it references a sub-class instead (like
409 // StaticIPAssignmentSettingData), its not a valid target
410 if ( ! ($instance.SettingData ISA CIM_IPAssignmentSettingData) ) {
411     $OperationError = smNewInstance("CIM_Error");
412     //CIM_ERR_FAILED
413     $OperationError.CIMStatusCode = 1;
414     //Software Error
415     $OperationError.ErrorType = 4;
416     //Unknown
417     $OperationError.PerceivedSeverity = 0;
418     $OperationError.OwningEntity = DMTF:SMCLP;
419     $OperationError.MessageID = 0x0000000E;
420     $OperationError.Message = "The \"IsCurrent\" property can not be modified for
421         this instance."
422     $OperationError.MessageArguments = { "IsCurrent" };
423     &smAddError($job, $OperationError);
424     &smMakeCommandStatus($job);
425     &smEnd;
426 }
427 // try to find the CIM_IPConfigurationService
428 $Services[] = smOpAssociators(
429     $instance.ManagedElement,
430     "CIM_ServiceAffectsElement",
431     "CIM_IPConfigurationService",
432     NULL,
433     NULL);
434 //no service associated, so applying configuration is not supported
435 if (NULL == Services[0]) {
436     $OperationError = smNewInstance("CIM_Error");
437     //CIM_ERR_FAILED
438     $OperationError.CIMStatusCode = 1;
439     //Software Error
440     $OperationError.ErrorType = 4;
441     //Unknown

```



```

442     $OperationError.PerceivedSeverity = 0;
443     $OperationError.OwningEntity = DMTF:SMCLP;
444     $OperationError.MessageID = 0x00000001;
445     $OperationError.Message = "Operation is not supported";
446     &smAddError($job, $OperationError);
447     &smMakeCommandStatus($job);
448     &smEnd;
449 }//
450 //Take the first instance we find
451 $Service-> = $Services[0].getObjectPath();
452 // if current configuration, force option is required
453 // value of 1 is "Is Current"
454 if ($instance.IsCurrent == 1 && !#force) {
455     $OperationError = smNewInstance("CIM_Error");
456     //CIM_ERR_FAILED
457     $OperationError.CIMStatusCode = 1;
458     //Software Error
459     $OperationError.ErrorType = 4;
460     //Unknown
461     $OperationError.PerceivedSeverity = 0;
462     $OperationError.OwningEntity = DMTF:SMCLP;
463     $OperationError.MessageID = 0x0000000F;
464     $OperationError.Message = "The selected configuration is already active.
465     Use the force option to re-apply it.";
466     &smAddError($job, $OperationError);
467     &smMakeCommandStatus($job);
468     &smEnd;
469 }
470 //invoke the method
471 //Step 6, build parameter lists for method invocation
472 %InArguments[] = {newArgument("Endpoint", $instance.ManagedElement),
473 newArgument ("Configuration", $instance.SettingData),
474 %OutArguments[] = { newArgument("Job",
475     instanceConcreteJob.getObjectPath() )};
476 //step 7, invoke method
477 #returnStatus = smOpInvokeMethod ($Service->,
478     "ApplySettingToIPProtocolEndpoint",
479     %InArguments[],
480     %OutArguments[]);
481 //step 8, process return code to CLP Command Status
482 if (0 != #Error.code) {
483     //method invocation failed
484     if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
485         //if the method invocation contains an embedded error
486         //use it for the Error for the overall job
487         &smAddError($job, #Error.$error[0]);
488         &smMakeCommandStatus($job);
489         &smEnd;
490     }
491     else if ( 17 == #Error.code ) {

```

```

492     //17 - CIM_ERR_METHOD_NOT_FOUND
493     // The specified extrinsic method does not exist.
494     $OperationError = smNewInstance("CIM_Error");
495     // CIM_ERR_METHOD_NOT_FOUND
496     $OperationError.CIMStatusCode = 17;
497     //Software Error
498     $OperationError.ErrorType = 10;
499     //Unknown
500     $OperationError.PerceivedSeverity = 0;
501     $OperationError.OwningEntity = DMTF:SMCLP;
502     $OperationError.MessageID = 0x00000001;
503     $OperationError.Message = "Operation is not supported."
504     &smAddError($job, $OperationError);
505     &smMakeCommandStatus($job);
506     &smEnd;
507 }
508 else {
509     //operation failed, but no detailed error instance, need to make one up
510     //make an Error instance and associate with job for Operation
511     $OperationError = smNewInstance("CIM_Error");
512     //CIM_ERR_FAILED
513     $OperationError.CIMStatusCode = 1;
514     //Software Error
515     $OperationError.ErrorType = 4;
516     //Unknown
517     $OperationError.PerceivedSeverity = 0;
518     $OperationError.OwningEntity = DMTF:SMCLP;
519     $OperationError.MessageID = 0x00000009;
520     $OperationError.Message = "An internal software error has occurred.";
521     &smAddError($job, $OperationError);
522     &smMakeCommandStatus($job);
523     &smEnd;
524 }
525 }//if CIM op failed
526 else if (0 == #returnStatus) {
527     //completed successfully
528     &smCommandCompleted($job);
529     &smEnd;
530 }
531 else if (4096 == #returnStatus) {
532     //job spawned, need to watch for it to finish
533     //while the jobstate is "Running"
534     while (4 == $instanceConcreteJob.JobState){<busy wait>}
535     if (2 != $job.OperationalStatus) {
536         %InArguments[] = { }
537         %OutArguments[] = {newArgument("Job", $instanceConcreteJob.getObjectPath())}
538         #Error = smOpInvokeMethod($job,
539             "GetError"
540             %InArguments,

```

```

541         %OutArguments,
542         #returncode);
543     //Method invocation failed, internal processing error
544     if ( (0 != #Error.code) || (0 != #returncode) ) {
545     //make an Error instance and associate with job for Operation
546         $OperationError = smNewInstance("CIM_Error");
547         //CIM_ERR_FAILED
548         $OperationError.CIMStatusCode = 1;
549         //Software Error
550         $OperationError.ErrorType = 4;
551         //Unknown
552         $OperationError.PerceivedSeverity = 0;
553         $OperationError.OwningEntity = DMTF:SMCLP;
554         $OperationError.MessageID = 0x00000009;
555         $OperationError.Message = "An internal software error has occurred.";
556         &smAddError($job, $OperationError);
557         &smMakeCommandStatus($job);
558         &smEnd;
559     }
560     else {
561         //make command status
562         $joberror = %OutArguments["Error"];
563         &smMakeCommandExecutionFailed($job, {$joberror});
564     } //end if have CIM_Error from GetError()
565 } //embedded job not OK
566 }
567 else {
568     //unspecified return code, generic failure
569     $OperationError = smNewInstance("CIM_Error");
570     //CIM_ERR_FAILED
571     $OperationError.CIMStatusCode = 1;
572     //Other
573     $OperationError.ErrorType = 1;
574     //Low
575     $OperationError.PerceivedSeverity = 2;
576     $OperationError.OwningEntity = DMTF:SMCLP;
577     $OperationError.MessageID = 0x00000002;
578     $OperationError.Message = "Failed. No further information is available.";
579     &smAddError($job, $OperationError);
580     &smMakeCommandStatus($job);
581     &smEnd;
582 }

```

### 583 6.3.3 Show

584 This section describes how to implement the `show` verb when applied to an instance of  
585 `CIM_ElementSettingData`. Implementations shall support the use of the `show` verb with  
586 `CIM_ElementSettingData`.

587 The `show` command is used to display information about the `CIM_ElementSettingData` instance or  
588 instances.

### 589 **6.3.3.1 Show Multiple Instances – CIM\_IPProtocolEndpoint Reference with Other Reference to** 590 **CIM\_IPAssignmentSettingData**

591 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
592 to a `show` command issued against `CIM_ElementSettingData` where only one reference is specified and  
593 the reference is to an instance of `CIM_IPProtocolEndpoint`.

#### 594 **6.3.3.1.1 Command Form**

```
595 show <CIM_ElementSettingData multiple objects>
```

#### 596 **6.3.3.1.2 CIM Requirements**

597 See `CIM_ElementSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
598 mandatory properties.

#### 599 **6.3.3.1.3 Behavior Requirements**

##### 600 **6.3.3.1.3.1 Preconditions**

601 `$instance` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
602 `CIM_ElementSettingData`.

603 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

##### 604 **6.3.3.1.3.2 Pseudo Code**

```
605 #propertylist[] = NULL;
606 if (#all == false)
607 {
608     #propertylist[] = { "IsCurrent", "IsDefault", "IsNext" };
609 }
610 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),
611     #propertylist[] );
612 &smEnd;
```

### 613 **6.3.3.2 Show Multiple Instances – CIM\_IPAssignmentSettingData Reference**

614 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
615 to a `show` command issued against `CIM_ElementSettingData` where only one reference is specified and  
616 the reference is to an instance of `CIM_IPAssignmentSettingData`.

#### 617 **6.3.3.2.1 Command Form**

```
618 show <CIM_ElementSettingData multiple objects>
```

#### 619 **6.3.3.2.2 CIM Requirements**

620 See `CIM_ElementSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
621 mandatory properties.

### 622 6.3.3.2.3 Behavior Requirements

#### 623 6.3.3.2.3.1 Preconditions

624 \$instance contains the instance of CIM\_IPAssignmentSettingData which is referenced by  
625 CIM\_ElementSettingData.

626 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

#### 627 6.3.3.2.3.2 Pseudo Code

```
628 #propertylist[] = NULL;
629 if (#all == false)
630 {
631     #propertylist[] = { "IsCurrent", "IsDefault", "IsNext" };
632 }
633 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),
634     #propertylist[] );
635 &smEnd;
```

### 636 6.3.3.3 Show Multiple Instances – CIM\_IPProtocolEndpoint Reference with Other Reference to 637 CIM\_StaticIPAssignmentSettingData

638 This command form is for the show verb applied to multiple instances. This command form corresponds  
639 to a show command issued against CIM\_ElementSettingData where only one reference is specified and  
640 the reference is to an instance of CIM\_IPProtocolEndpoint.

#### 641 6.3.3.3.1 Command Form

```
642 show <CIM_ElementSettingData multiple objects>
```

#### 643 6.3.3.3.2 CIM Requirements

644 See CIM\_ElementSettingData in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
645 mandatory properties.

### 646 6.3.3.3.3 Behavior Requirements

#### 647 6.3.3.3.3.1 Preconditions

648 \$instance contains the instance of CIM\_IPProtocolEndpoint which is referenced by  
649 CIM\_ElementSettingData.

650 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

#### 651 6.3.3.3.3.2 Pseudo Code

```
652 #propertylist[] = NULL;
653 if (#all == false)
654 {
655     #propertylist[] = { "IsCurrent" };
656 }
657 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),
658     #propertylist[] );
659 &smEnd;
```

### 660 **6.3.3.4 Show Multiple Instances – CIM\_StaticIPAssignmentSettingData Reference**

661 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
662 to a `show` command issued against `CIM_ElementSettingData` where only one reference is specified and  
663 the reference is to an instance of `CIM_StaticIPAssignmentSettingData`.

#### 664 **6.3.3.4.1 Command Form**

```
665 show <CIM_ElementSettingData multiple objects>
```

#### 666 **6.3.3.4.2 CIM Requirements**

667 See `CIM_ElementSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
668 mandatory properties.

#### 669 **6.3.3.4.3 Behavior Requirements**

##### 670 **6.3.3.4.3.1 Preconditions**

671 `$instance` contains the instance of `CIM_StaticIPAssignmentSettingData` which is referenced by  
672 `CIM_ElementSettingData`.

673 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

##### 674 **6.3.3.4.3.2 Pseudo Code**

```
675 #propertylist[] = NULL;
676 if (#all == false)
677 {
678     #propertylist[] = { "IsCurrent" };
679 }
680 &smShowAssociationInstances ( "CIM_ElementSettingData", $instance.getObjectPath(),
681     #propertylist[] );
682 &smEnd;
```

### 683 **6.3.3.5 Show a Single Instance – CIM\_IPProtocolEndpoint and CIM\_IPAssignmentSettingData** 684 **References**

685 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
686 a `show` command issued against `CIM_ElementSettingData` where both references are specified and  
687 therefore the desired instance is unambiguously identified.

#### 688 **6.3.3.5.1 Command Form**

```
689 show <CIM_ElementSettingData single object>
```

#### 690 **6.3.3.5.2 CIM Requirements**

691 See `CIM_ElementSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
692 mandatory properties.

#### 693 **6.3.3.5.3 Behavior Requirements**

##### 694 **6.3.3.5.3.1 Preconditions**

695 `$instanceA` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
696 `CIM_ElementSettingData`.

697 \$instanceB contains the instance of CIM\_IPAssignmentSettingData referenced by  
698 CIM\_ElementSettingData.

699 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

#### 700 **6.3.3.5.3.2 Pseudo Code**

```
701 #propertylist[] = NULL;
702 if (#all == false)
703 {
704     #propertylist[] = { "IsCurrent", "IsDefault", "IsNext" };
705 }
706 &smShowAssociationInstance ( "CIM_ElementSettingData", $instanceA.getObjectPath(),
707     $instanceB.getObjectPath(), #propertylist[] );
708 &smEnd;
```

#### 709 **6.3.3.6 Show a Single Instance – CIM\_IPProtocolEndpoint and 710 CIM\_StaticIPAssignmentSettingData References**

711 This command form is for the show verb applied to a single instance. This command form corresponds to  
712 a show command issued against CIM\_ElementSettingData where both references are specified and  
713 therefore the desired instance is unambiguously identified.

##### 714 **6.3.3.6.1 Command Form**

```
715 show <CIM_ElementSettingData single object>
```

##### 716 **6.3.3.6.2 CIM Requirements**

717 See CIM\_ElementSettingData in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
718 mandatory properties.

##### 719 **6.3.3.6.3 Behavior Requirements**

###### 720 **6.3.3.6.3.1 Preconditions**

721 \$instanceA contains the instance of CIM\_IPProtocolEndpoint which is referenced by  
722 CIM\_ElementSettingData.

723 \$instanceB contains the instance of CIM\_StaticIPAssignmentSettingData referenced by  
724 CIM\_ElementSettingData.

725 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

###### 726 **6.3.3.6.3.2 Pseudo Code**

```
727 #propertylist[] = NULL;
728 if (#all == false)
729 {
730     #propertylist[] = { "IsCurrent" };
731 }
732 &smShowAssociationInstance ( "CIM_ElementSettingData", $instanceA.getObjectPath(),
733     $instanceB.getObjectPath(), #propertylist[] );
734 &smEnd;
```

735 **6.4 CIM\_EnabledLogicalElementCapabilities**736 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

737 Table 4 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
 738 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 739 verb and target. Table 4 is for informational purposes only; in case of a conflict between Table 4 and  
 740 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 741 information in Table 4.

742 **Table 4 – Command Verb Requirements for CIM\_EnabledLogicalElementCapabilities**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.4.2.
start	Not supported	
stop	Not supported	

743 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 744 `reset`, `set`, `start`, and `stop`.

745 **6.4.1 Ordering of Results**

746 When results are returned for multiple instances of `CIM_EnabledLogicalElementCapabilities`,  
 747 implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

- 748 • Results for `CIM_EnabledLogicalElementCapabilities` are unordered; therefore, no algorithm is  
 749 defined.

750 **6.4.2 Show**

751 This section describes how to implement the `show` verb when applied to an instance of  
 752 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with  
 753 `CIM_EnabledLogicalElementCapabilities`.

754 The `show` verb is used to display information about the capabilities.

755 **6.4.2.1 Show a Single Instance**

756 This command form is for the `show` verb applied to a single instance of  
 757 `CIM_EnabledLogicalElementCapabilities`.

758 **6.4.2.1.1 Command Form**

759 `show <CIM_EnabledLogicalElementCapabilities single object>`



### 760 6.4.2.1.2 CIM Requirements

761 See CIM\_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [IP Interface Profile](#) for  
762 the list of mandatory properties.

### 763 6.4.2.1.3 Behavior Requirements

#### 764 6.4.2.1.3.1 Preconditions

765 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

#### 766 6.4.2.1.3.2 Pseudo Code

```
767 $instance=<CIM_EnabledLogicalElementCapabilities single object>
768 #propertylist[] = NULL;
769 if (false == #all)
770     {
771         #propertylist[] = {/all non-key properties};
772     }
773 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
774 &smEnd;
```

### 775 6.4.2.2 Show Multiple Instances

776 This command form is for the show verb applied to multiple instances of  
777 CIM\_EnabledLogicalElementCapabilities. This command form corresponds to UFST-based selection  
778 within a capabilities collection.

#### 779 6.4.2.2.1 Command Form

```
780 show <CIM_EnabledLogicalElementCapabilities multiple objects>
```

#### 781 6.4.2.2.2 CIM Requirements

782 See CIM\_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [IP Interface Profile](#) for  
783 the list of mandatory properties.

#### 784 6.4.2.2.3 Behavior Requirements

##### 785 6.4.2.2.3.1 Preconditions

786 \$containerInstance contains the instance of CIM\_ConcreteCollection for which contained  
787 CIM\_Capabilities instances are displayed. CIM\_Capabilities instances are addressed via an aggregating  
788 instance of CIM\_ConcreteCollection.

789 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 790 6.4.2.2.3.2 Pseudo Code

```
791 #propertylist[] = NULL;
792 if (false == #all)
793     {
794         #propertylist[] = {/all non-key properties};
795     }
796 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
797     $containerInstance.getObjectPath(), #propertylist[] );
798 &smEnd;
```

## 799 6.5 CIM\_HostedAccessPoint

800 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

801 Table 5The following table lists each SM CLP verb, the required level of support for the verb in  
 802 conjunction with instances of the target class, and, when appropriate, a cross-reference to the section  
 803 detailing the mapping for the verb and target. Table 5 is for informational purposes only; in case of a  
 804 conflict between Table 5 and requirements detailed in the following sections, the text detailed in the  
 805 following sections supersedes the information in Table 5.

806 **Table 5 – Command Verb Requirements for CIM\_HostedAccessPoint**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.5.2.
start	Not supported	
stop	Not supported	

807 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 808 `reset`, `set`, `start`, and `stop`.

### 809 6.5.1 Ordering of Results

810 When results are returned for multiple instances of `CIM_HostedAccessPoint`, implementations shall utilize  
 811 the following algorithm to produce the natural (that is, default) ordering:

- 812 • Results for `CIM_HostedAccessPoint` are unordered; therefore, no algorithm is defined.

### 813 6.5.2 Show

814 This section describes how to implement the `show` verb when applied to an instance of  
 815 `CIM_HostedAccessPoint`. Implementations shall support the use of the `show` verb with  
 816 `CIM_HostedAccessPoint`.

817 The `show` command is used to display information about the `CIM_HostedAccessPoint` instance or  
 818 instances.

#### 819 6.5.2.1 Show Multiple Instances – CIM\_ComputerSystem Reference

820 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
 821 to a `show` command issued against `CIM_HostedAccessPoint` where only one reference is specified and  
 822 the reference is to an instance of `CIM_ComputerSystem`.

##### 823 6.5.2.1.1 Command Form

824 `show <CIM_HostedAccessPoint multiple objects>`

### 825 6.5.2.1.2 CIM Requirements

826 See CIM\_HostedAccessPoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
827 mandatory properties.

### 828 6.5.2.1.3 Behavior Requirements

#### 829 6.5.2.1.3.1 Preconditions

830 \$instance contains the instance of CIM\_ComputerSystem which is referenced by  
831 CIM\_HostedAccessPoint.

#### 832 6.5.2.1.3.2 Pseudo Code

```
833 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath() );  
834 &smEnd;
```

### 835 6.5.2.2 Show a Single Instance – CIM\_IPProtocolEndpoint Reference

836 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
837 a `show` command issued against CIM\_HostedAccessPoint where the reference specified is to an  
838 instance of CIM\_CIM\_IPProtocolEndpoint. An instance of CIM\_CIM\_IPProtocolEndpoint is referenced by  
839 exactly one instance of CIM\_HostedAccessPoint. Therefore, a single instance will be returned.

#### 840 6.5.2.2.1 Command Form

```
841 show <CIM_HostedAccessPoint single object>
```

#### 842 6.5.2.2.2 CIM Requirements

843 See CIM\_HostedAccessPoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
844 mandatory properties.

#### 845 6.5.2.2.3 Behavior Requirements

##### 846 6.5.2.2.3.1 Preconditions

847 \$instance contains the instance of CIM\_CIM\_IPProtocolEndpoint which is referenced by  
848 CIM\_HostedAccessPoint.

##### 849 6.5.2.2.3.2 Pseudo Code

```
850 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath() );  
851 &smEnd;
```

### 852 6.5.2.3 Show a Single Instance – CIM\_CIM\_RemoteServiceAccessPoint Reference

853 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
854 a `show` command issued against CIM\_HostedAccessPoint where the reference specified is to an  
855 instance of CIM\_CIM\_RemoteServiceAccessPoint. An instance of CIM\_CIM\_RemoteServiceAccessPoint  
856 is referenced by exactly one instance of CIM\_HostedAccessPoint. Therefore, a single instance will be  
857 returned.

#### 858 6.5.2.3.1 Command Form

```
859 show <CIM_HostedAccessPoint single object>
```

**860 6.5.2.3.2 CIM Requirements**

861 See CIM\_HostedAccessPoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
862 mandatory properties.

**863 6.5.2.3.3 Behavior Requirements****864 6.5.2.3.3.1 Preconditions**

865 \$instance contains the instance of CIM\_CIM\_RemoteServiceAccessPoint which is referenced by  
866 CIM\_HostedAccessPoint.

**867 6.5.2.3.3.2 Pseudo Code**

```
868 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath() );  
869 &smEnd;
```

**870 6.5.2.4 Show a Single Instance – Both References (IPProtocolEndpoint)**

871 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
872 a `show` command issued against CIM\_HostedAccessPoint where both references are specified and  
873 therefore the desired instance is unambiguously identified.

**874 6.5.2.4.1 Command Form**

```
875 show <CIM_HostedAccessPoint single object>
```

**876 6.5.2.4.2 CIM Requirements**

877 See CIM\_HostedAccessPoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
878 mandatory properties.

**879 6.5.2.4.3 Behavior Requirements****880 6.5.2.4.3.1 Preconditions**

881 \$instanceA contains the instance of CIM\_ComputerSystem which is referenced by  
882 CIM\_HostedAccessPoint.

883 \$instanceB contains the instance of CIM\_CIM\_IPProtocolEndpoint which is referenced by  
884 CIM\_HostedAccessPoint.

**885 6.5.2.4.3.2 Pseudo Code**

```
886 &smShowAssociationInstance ( "CIM_HostedAccessPoint", $instanceA.getObjectPath(),  
887     $instanceB.getObjectPath() );  
888 &smEnd;
```

**889 6.5.2.5 Show a Single Instance – Both References (RemoteServiceAccessPoint)**

890 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
891 a `show` command issued against CIM\_HostedAccessPoint where both references are specified and  
892 therefore the desired instance is unambiguously identified.

**893 6.5.2.5.1 Command Form**

```
894 show <CIM_HostedAccessPoint single object>
```

895 **6.5.2.5.2 CIM Requirements**

896 See CIM\_HostedAccessPoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
897 mandatory properties.

898 **6.5.2.5.3 Behavior Requirements**

899 **6.5.2.5.3.1 Preconditions**

900 \$instanceA contains the instance of CIM\_ComputerSystem which is referenced by  
901 CIM\_HostedAccessPoint.

902 \$instanceB contains the instance of CIM\_CIM\_RemoteServiceAccessPoint which is referenced by  
903 CIM\_HostedAccessPoint.

904 **6.5.2.5.3.2 Pseudo Code**

```
905 &smShowAssociationInstance ( "CIM_HostedAccessPoint", $instanceA.getObjectPath(),
906     $instanceB.getObjectPath() );
907 &smEnd;
```

908 **6.6 CIM\_HostedService**

909 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

910 Table 6 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
911 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
912 verb and target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and  
913 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
914 information in Table 6.

915 **Table 6 – Command Verb Requirements for CIM\_HostedService**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.6.2.
start	Not supported	
stop	Not supported	

916 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
917 `reset`, `set`, `start`, and `stop`.

918 **6.6.1 Ordering of Results**

919 When results are returned for multiple instances of CIM\_HostedService, implementations shall utilize the  
920 following algorithm to produce the natural (that is, default) ordering:

- 921 • Results for CIM\_HostedService are unordered; therefore, no algorithm is defined.

## 922 6.6.2 Show

923 This section describes how to implement the `show` verb when applied to an instance of  
924 `CIM_HostedService`. Implementations shall support the use of the `show` verb with `CIM_HostedService`.

925 The `show` command is used to display information about the `CIM_HostedService` instance or instances.

### 926 6.6.2.1 Show Multiple Instances – CIM\_ComputerSystem Reference

927 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
928 to a `show` command issued against `CIM_HostedService` where only one reference is specified and the  
929 reference is to an instance of `CIM_ComputerSystem`.

#### 930 6.6.2.1.1 Command Form

```
931 show <CIM_HostedService multiple objects>
```

#### 932 6.6.2.1.2 CIM Requirements

933 See `CIM_HostedService` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of mandatory  
934 properties.

#### 935 6.6.2.1.3 Behavior Requirements

##### 936 6.6.2.1.3.1 Preconditions

937 `$instance` contains the instance of `CIM_ComputerSystem` which is referenced by `CIM_HostedService`.

##### 938 6.6.2.1.3.2 Pseudo Code

```
939 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );  
940 &smEnd;
```

### 941 6.6.2.2 Show a Single Instance – CIM\_IPConfigurationService Reference

942 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
943 a `show` command issued against `CIM_HostedService` where the reference specified is to an instance of  
944 `CIM_IPConfigurationService`. An instance of `CIM_IPConfigurationService` is referenced by exactly one  
945 instance of `CIM_HostedService`. Therefore, a single instance will be returned.

#### 946 6.6.2.2.1 Command Form

```
947 show <CIM_HostedService single object>
```

#### 948 6.6.2.2.2 CIM Requirements

949 See `CIM_HostedService` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of mandatory  
950 properties.

#### 951 6.6.2.2.3 Behavior Requirements

##### 952 6.6.2.2.3.1 Preconditions

953 `$instance` contains the instance of `CIM_IPConfigurationService` which is referenced by  
954 `CIM_HostedService`.

955 **6.6.2.2.3.2 Pseudo Code**

```
956 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );
957 &smEnd;
```

958 **6.6.2.3 Show a Single Instance – Both References**

959 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 960 a `show` command issued against `CIM_HostedService` where both references are specified and therefore  
 961 the desired instance is unambiguously identified.

962 **6.6.2.3.1 Command Form**

```
963 show <CIM_HostedService single object>
```

964 **6.6.2.3.2 CIM Requirements**

965 See `CIM_HostedService` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of mandatory  
 966 properties.

967 **6.6.2.3.3 Behavior Requirements**

968 **6.6.2.3.3.1 Preconditions**

969 `$instanceA` contains the instance of `CIM_ComputerSystem` which is referenced by  
 970 `CIM_HostedService`.

971 `$instanceB` contains the instance of `CIM_IPConfigurationService` which is referenced by  
 972 `CIM_HostedService`.

973 **6.6.2.3.3.2 Pseudo Code**

```
974 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
975     $instanceB.getObjectPath() );
976 &smEnd;
```

977 **6.7 CIM\_IPAssignmentSettingData**

978 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

979 Table 7 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
 980 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 981 verb and target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and  
 982 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 983 information in Table 7.

984 **Table 7 – Command Verb Requirements for CIM\_IPAssignmentSettingData**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.7.2.

Command Verb	Requirement	Comments
show	Shall	See 6.7.3.
start	Not supported	
stop	Not supported	

985 No mapping is defined for the following verbs for the specified target: *create*, *delete*, *dump*, *load*,  
 986 *reset*, *start*, and *stop*.

### 987 6.7.1 Ordering of Results

988 When results are returned for multiple instances of *CIM\_IPAssignmentSettingData*, implementations shall  
 989 utilize the following algorithm to produce the natural (that is, default) ordering:

- 990 • Results for *CIM\_IPAssignmentSettingData* are unordered; therefore, no algorithm is defined.

### 991 6.7.2 Set

992 This section describes how to implement the *set* verb when it is applied to an instance of  
 993 *CIM\_IPAssignmentSettingData*. Implementations may support the use of the *set* verb with  
 994 *CIM\_IPAssignmentSettingData*.

995 The *set* verb is used to modify descriptive properties of the *CIM\_IPAssignmentSettingData* instance.

#### 996 6.7.2.1 General Usage of Set for a Single Property

997 This command form corresponds to the general usage of the *set* verb to modify a single property of a  
 998 target instance. This is the most common case.

999 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1000 the requirement for supporting modification of the property using the *ModifyInstance* operation as defined  
 1001 in the [IP Interface Profile](#).

##### 1002 6.7.2.1.1 Command Form

```
1003 set <CIM_IPAssignmentSettingData single instance> <propertyname>=<propertyvalue>
```

##### 1004 6.7.2.1.2 CIM Requirements

1005 See *CIM\_IPAssignmentSettingData* in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1006 modifiable properties.

##### 1007 6.7.2.1.3 Behavior Requirements

```
1008 $instance=<CIM_IPAssignmentSettingData single instance>
1009 #propertyNames[] = {<propertyname>};
1010 #propertyValues[] = {<propertyvalue>};
1011 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1012 &smEnd;
```

#### 1013 6.7.2.2 General Usage of Set for Multiple Properties

1014 This command form corresponds to the general usage of the *set* verb to modify multiple properties of a  
 1015 target instance where there is not an explicit relationship between the properties. This is the most  
 1016 common case.



1017 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1018 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
 1019 in the [IP Interface Profile](#).

#### 1020 6.7.2.2.1 Command Form

```
1021 set <CIM_IPAssignmentSettingData single instance> <propertyname1>=<propertyvalue1>  
1022 <propertynamen>=<propertyvaluen>
```

#### 1023 6.7.2.2.2 CIM Requirements

1024 See CIM\_IPAssignmentSettingData in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1025 mandatory properties.

#### 1026 6.7.2.2.3 Behavior Requirements

```
1027 $instance=<CIM_IPAssignmentSettingData single instance>  
1028 #propertyNames[] = {<propertyname>};  
1029 for #i < n  
1030 {  
1031     #propertyNames[#i] = <propertyname#i>  
1032     #propertyValues[#i] = <propertyvalue#i>  
1033 }  
1034 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );  
1035 &smEnd;
```

### 1036 6.7.3 Show

1037 This section describes how to implement the `show` verb when applied to an instance of  
 1038 CIM\_IPAssignmentSettingData. Implementations shall support the use of the `show` verb with  
 1039 CIM\_IPAssignmentSettingData.

1040 The `show` verb is used to display information about CIM\_IPAssignmentSettingData.

#### 1041 6.7.3.1 Show a Single Instance

1042 This command form is for the `show` verb applied to a single instance of CIM\_IPAssignmentSettingData.

##### 1043 6.7.3.1.1 Command Form

```
1044 show <CIM_IPAssignmentSettingData single object>
```

##### 1045 6.7.3.1.2 CIM Requirements

1046 See CIM\_IPAssignmentSettingData in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1047 mandatory properties.

##### 1048 6.7.3.1.3 Behavior Requirements

###### 1049 6.7.3.1.3.1 Preconditions

1050 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1051 **6.7.3.1.3.2 Pseudo Code**

```

1052 $instance=<CIM_IPAssignmentSettingData single object>
1053 #propertylist[] = NULL;
1054 if (false == #all)
1055     {
1056         #propertylist[] = { "ElementName", "AddressOrigin" }
1057     }
1058 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1059 &smEnd;

```

1060 **6.7.3.2 Show Multiple Instances**

1061 This command form is for the `show` verb applied to multiple instances of `CIM_IPAssignmentSettingData`.  
 1062 This command form corresponds to UFT-based selection within a scoping system.

1063 **6.7.3.2.1 Command Form**

```

1064 show <CIM_IPAssignmentSettingData multiple objects>

```

1065 **6.7.3.2.2 CIM Requirements**

1066 See `CIM_IPAssignmentSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1067 mandatory properties.

1068 **6.7.3.2.3 Behavior Requirements**1069 **6.7.3.2.3.1 Preconditions**

1070 `$containerInstance` contains the instance of `CIM_ConcreteCollection` for which related  
 1071 `CIM_IPAssignmentSettingData` instances are displayed. SM ME Addressing requires that the  
 1072 `CIM_IPAssignmentSettingData` instance be associated with an instance of `CIM_ConcreteCollection` via  
 1073 an instance of the `CIM_MemberOfCollection` association.

1074 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1075 **6.7.3.2.3.2 Pseudo Code**

```

1076 #propertylist[] = NULL;
1077 if (false == #all)
1078     {
1079         #propertylist[] = { "ElementName", "AddressOrigin" }
1080     }
1081 &smShowInstances ( "CIM_IPAssignmentSettingData", "CIM_OrderedComponent",
1082     $containerInstance.getObjectPath(), #propertylist[] );
1083 &smEnd;

```

1084 **6.8 CIM\_IPConfigurationService**

1085 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1086 Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
 1087 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 1088 verb and target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and  
 1089 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1090 information in Table 8.

1091

**Table 8 – Command Verb Requirements for CIM\_IPConfigurationService**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.8.2.
show	Shall	See 6.8.3.
start	Not supported	
stop	Not supported	

1092 No mapping is defined for the following verbs for the specified target: *create*, *delete*, *dump*, *load*,  
 1093 *reset*, *start*, and *stop*.

### 1094 **6.8.1 Ordering of Results**

1095 When results are returned for multiple instances of CIM\_IPConfigurationService, implementations shall  
 1096 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1097 • Results for CIM\_IPConfigurationService are unordered; therefore, no algorithm is defined.

### 1098 **6.8.2 Set**

1099 This section describes how to implement the *set* verb when applied to an instance of  
 1100 CIM\_IPConfigurationService. Implementations may support the use of the *set* verb with  
 1101 CIM\_IPConfigurationService.

1102 No properties of the CIM\_IPConfigurationService instance are writeable via the intrinsic ModifyInstance  
 1103 operation. Therefore, the only command form specified is for requesting a state change on the instance  
 1104 via assignment to the RequestedState property.

#### 1105 **6.8.2.1 General Usage of Set for a Single Property**

1106 This command form corresponds to the general usage of the *set* verb to modify a single property of a  
 1107 target instance. This is the most common case.

1108 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1109 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
 1110 in the [IP Interface Profile](#).

##### 1111 **6.8.2.1.1 Command Form**

```
1112 set <CIM_IPConfigurationService single instance> <propertyname>=<propertyvalue>
```

##### 1113 **6.8.2.1.2 CIM Requirements**

1114 See CIM\_IPConfigurationService in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1115 modifiable properties.

### 1116 6.8.2.1.3 Behavior Requirements

```

1117 $instance=<CIM_IPConfigurationService single instance>
1118 #propertyName[] = {<propertyname>};
1119 #propertyValues[] = {<propertyvalue>};
1120 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1121 &smEnd;

```

### 1122 6.8.2.2 General Usage of Set for Multiple Properties

1123 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
 1124 target instance where there is not an explicit relationship between the properties. This is the most  
 1125 common case.

1126 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1127 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
 1128 in the [IP Interface Profile](#).

#### 1129 6.8.2.2.1 Command Form

```

1130 set <CIM_IPConfigurationService single instance> <propertyname1>=<propertyvalue1>
1131 <propertynamen>=<propertyvaluen>

```

#### 1132 6.8.2.2.2 CIM Requirements

1133 See `CIM_IPConfigurationService` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1134 mandatory properties.

#### 1135 6.8.2.2.3 Behavior Requirements

```

1136 $instance=<CIM_IPConfigurationService single instance>
1137 #propertyName[] = {<propertyname>};
1138 for #i < n
1139 {
1140     #propertyName[#i] = <propertyname#i>
1141     #propertyValues[#i] = <propertyvalue#i>
1142 }
1143 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1144 &smEnd;

```

### 1145 6.8.3 Show

1146 This section describes how to implement the `show` verb when applied to an instance of  
 1147 `CIM_IPConfigurationService`. Implementations shall support the use of the `show` verb with  
 1148 `CIM_IPConfigurationService`.

1149 The `show` verb is used to display information about the `CIM_IPConfigurationService` instance.

#### 1150 6.8.3.1 Show a Single Instance

1151 This command form is for the `show` verb applied to a single instance of `CIM_IPConfigurationService`.

##### 1152 6.8.3.1.1 Command Form

```

1153 show <CIM_IPConfigurationService single object>

```

### 1154 6.8.3.1.2 CIM Requirements

1155 See CIM\_IPConfigurationService in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1156 mandatory properties.

### 1157 6.8.3.1.3 Behavior Requirements

```
1158 $instance=<CIM_IPConfigurationService single object>
1159 #propertylist[] = NULL;
1160 if (false == #all)
1161 {
1162     #propertylist[] = { "ElementName" }
1163 }
1164 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1165 &smEnd;
```

### 1166 6.8.3.2 Show Multiple Instances

1167 This command form is for the `show` verb applied to multiple instances of CIM\_IPConfigurationService.  
1168 This command form corresponds to UFT-based selection within a scoping system.

#### 1169 6.8.3.2.1 Command Form

```
1170 show <CIM_IPConfigurationService multiple objects>
```

#### 1171 6.8.3.2.2 CIM Requirements

1172 See CIM\_IPConfigurationService in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1173 mandatory properties.

#### 1174 6.8.3.2.3 Behavior Requirements

##### 1175 6.8.3.2.3.1 Preconditions

1176 `$containerInstance` contains the instance of CIM\_ComputerSystem for which scoped  
1177 CIM\_IPConfigurationService instances are displayed. The [IP Interface Profile](#) requires that the  
1178 CIM\_IPConfigurationService instance be associated with its scoping system via an instance of the  
1179 CIM\_HostedService association.

##### 1180 6.8.3.2.3.2 Pseudo Code

```
1181 if (false == #all)
1182 {
1183     #propertylist[] = { "ElementName" }
1184 }
1185 &smShowInstances ( "CIM_IPConfigurationService", "CIM_HostedService",
1186     $containerInstance.getObjectPath(), #propertylist[] );
1187 &smEnd;
```

## 1188 6.9 CIM\_IPProtocolEndpoint

1189 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1190 Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with instances of  
1191 the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
1192 verb and target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and  
1193 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
1194 information in Table 9.

1195

Table 9 – Command Verb Requirements for CIM\_IPProtocolEndpoint

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	May	See 6.9.2.
set	May	See 6.9.3.
show	Shall	See 6.9.3.2.4.
start	May	See 6.9.4.
stop	May	See 6.9.5.

1196 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, and `load`.

### 1197 6.9.1 Ordering of Results

1198 When results are returned for multiple instances of `CIM_IPProtocolEndpoint`, implementations shall utilize  
1199 the following algorithm to produce the natural (that is, default) ordering:

- 1200 • Results for `CIM_IPProtocolEndpoint` are unordered; therefore, no algorithm is defined.

### 1201 6.9.2 Reset

1202 This section describes how to implement the `reset` verb when applied to an instance of  
1203 `CIM_IPProtocolEndpoint`. Implementations may support the use of the `reset` verb with  
1204 `CIM_IPProtocolEndpoint`.

1205 The `reset` verb is used to initiate a reset of the `CIM_IPProtocolEndpoint`.

#### 1206 6.9.2.1 Reset a Single Instance

1207 This command form is for the initiation of a reset action against a single IP interface. The mapping is  
1208 implemented as an invocation of the `RequestStateChange()` method on the instance.

##### 1209 6.9.2.1.1 Command Form

```
1210 reset <CIM_IPProtocolEndpoint single object>
```

##### 1211 6.9.2.1.2 CIM Requirements

```
1212 uint16 EnabledState;
1213 uint16 RequestedState;
1214 uint32 EnabledLogicalElement.RequestStateChange (
1215     [IN] uint16 RequestedState,
1216     [OUT] REF CIM_ConcreteJob Job,
1217     [IN] datetime TimeoutPeriod );
```

##### 1218 6.9.2.1.3 Behavior Requirements

```
1219 $instance=<CIM_IPProtocolEndpoint single object>
1220 smResetRSC ( $instance.getObjectPath() );
1221 &smEnd;
```

### 1222 6.9.3 Set

1223 This section describes how to implement the `set` verb when it is applied to an instance of  
1224 CIM\_IPProtocolEndpoint. Implementations may support the use of the `set` verb with  
1225 CIM\_IPProtocolEndpoint.

1226 The `set` verb is used to modify descriptive properties of the CIM\_IPProtocolEndpoint instance.

#### 1227 6.9.3.1 General Usage of Set for a Single Property

1228 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
1229 target instance. This is the most common case.

1230 The requirement for supporting modification of a property using this command form shall be equivalent to  
1231 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
1232 in the [IP Interface Profile](#).

##### 1233 6.9.3.1.1 Command Form

```
1234 set <CIM_IPProtocolEndpoint single instance> <propertyname>=<propertyvalue>
```

##### 1235 6.9.3.1.2 CIM Requirements

1236 See CIM\_IPProtocolEndpoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1237 modifiable properties.

##### 1238 6.9.3.1.3 Behavior Requirements

```
1239 $instance=<CIM_IPProtocolEndpoint single instance>
1240 #propertyNames[] = {<propertyname>};
1241 #propertyValues[] = {<propertyvalue>};
1242 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1243 &smEnd;
```

#### 1244 6.9.3.2 General Usage of Set for Multiple Properties

1245 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
1246 target instance where there is not an explicit relationship between the properties. This is the most  
1247 common case.

1248 The requirement for supporting modification of a property using this command form shall be equivalent to  
1249 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
1250 in the [IP Interface Profile](#).

##### 1251 6.9.3.2.1 Command Form

```
1252 set <CIM_IPProtocolEndpoint single instance> <propertyname1>=<propertyvalue1>
1253 <propertynamen>=<propertyvaluen>
```

##### 1254 6.9.3.2.2 CIM Requirements

1255 See CIM\_IPProtocolEndpoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1256 mandatory properties.

1257 **6.9.3.2.3 Behavior Requirements**

```

1258 $instance=<CIM_IPProtocolEndpoint single instance>
1259 #propertyName[] = {<propertyname>};
1260 for #i < n
1261     {
1262         #propertyName[#i] = <propertyname#i>
1263         #propertyValue[#i] = <propertyvalue#i>
1264     }
1265 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );
1266 &smEnd;

```

1267 **6.9.3.2.4 Show**

1268 This section describes how to implement the `show` verb when applied to an instance of  
 1269 CIM\_IPProtocolEndpoint. Implementations shall support the use of the `show` verb with  
 1270 CIM\_IPProtocolEndpoint.

1271 The `show` verb is used to display information about the IP interface.

1272 **6.9.3.3 Show a Single Instance**

1273 This command form is for the `show` verb applied to a single instance of CIM\_IPProtocolEndpoint.

1274 **6.9.3.3.1 Command Form**

```

1275 show <CIM_IPProtocolEndpoint single object>

```

1276 **6.9.3.3.2 CIM Requirements**

1277 See CIM\_IPProtocolEndpoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1278 mandatory properties.

1279 **6.9.3.3.3 Behavior Requirements**1280 **6.9.3.3.3.1 Preconditions**

1281 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1282 **6.9.3.3.3.2 Pseudo Code**

```

1283 $instance=<CIM_IPProtocolEndpoint single object>
1284 #propertylist[] = NULL;
1285 if (false == #all)
1286     {
1287         #propertylist[] = { "IPv4Address", "SubnetMask", "AddressOrigin" };
1288     }
1289 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1290 &smEnd;

```

1291 **6.9.3.4 Show Multiple Instances**

1292 This command form is for the `show` verb applied to multiple instances of CIM\_IPProtocolEndpoint. This  
 1293 command form corresponds to UFT-based selection within a scoping system.



1294 **6.9.3.4.1 Command Form**1295 `show <CIM_IPProtocolEndpoint multiple objects>`1296 **6.9.3.4.2 CIM Requirements**1297 See CIM\_IPProtocolEndpoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1298 mandatory properties.1299 **6.9.3.4.3 Behavior Requirements**1300 **6.9.3.4.3.1 Preconditions**1301 \$containerInstance contains the instance of CIM\_ComputerSystem for which scoped  
1302 CIM\_IPProtocolEndpoint instances are displayed. The [IP Interface Profile](#) requires that the  
1303 CIM\_IPProtocolEndpoint instance be associated with its scoping system via an instance of the  
1304 CIM\_HostedAccessPoint association.

1305 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1306 **6.9.3.4.3.2 Pseudo Code**1307 

```
#propertylist[] = NULL;
1308 if (false == #all)
1309     {
1310         #propertylist[] = { "IPv4Address", "SubnetMask", "AddressOrigin" };
1311     }
1312 &smShowInstances ( "CIM_IPProtocolEndpoint", "CIM_HostedAccessPoint",
1313     $containerInstance.getObjectPath(), #propertylist[] );
1314 &smEnd;
```

1315 **6.9.4 Start**1316 This section describes how to implement the `start` verb when applied to an instance of  
1317 CIM\_IPProtocolEndpoint. Implementations may support the use of the `start` verb with  
1318 CIM\_IPProtocolEndpoint.1319 The `start` verb is used to enable an IP interface.1320 **6.9.4.1 Start a Single Instance**1321 This command form is for the `start` verb applied to a single instance of CIM\_IPProtocolEndpoint.1322 **6.9.4.1.1 Command Form**1323 `start <CIM_IPProtocolEndpoint single object>`1324 **6.9.4.1.2 CIM Requirements**1325 

```
uint16 EnabledState;
1326 uint16 RequestedState;
1327 uint32 EnabledLogicalElement.RequestStateChange (
1328     [IN] uint16 RequestedState,
1329     [OUT] REF CIM_ConcreteJob Job,
1330     [IN] datetime TimeoutPeriod );
```

1331 **6.9.4.1.3 Behavior Requirements**

```
1332 $instance=<CIM_IPProtocolEndpoint single object>
1333 smStartRSC ( $instance.GetObjectPath() );
1334 &smEnd;
```

1335 **6.9.5 Stop**

1336 This section describes how to implement the `stop` verb when applied to an instance of  
1337 `CIM_IPProtocolEndpoint`. Implementations may support the use of the `stop` verb with  
1338 `CIM_IPProtocolEndpoint`.

1339 The `stop` verb is used to disable an IP interface.

1340 **6.9.5.1 Stop a Single Instance**

1341 This command form is for the `stop` verb applied to a single instance of `CIM_IPProtocolEndpoint`.

1342 **6.9.5.1.1 Command Form**

```
1343 stop <CIM_IPProtocolEndpoint single object>
```

1344 **6.9.5.1.2 CIM Requirements**

```
1345 uint16 EnabledState;
1346 uint16 RequestedState;
1347 uint32 EnabledLogicalElement.RequestStateChange (
1348     [IN] uint16 RequestedState,
1349     [OUT] REF CIM_ConcreteJob Job,
1350     [IN] datetime TimeoutPeriod );
```

1351 **6.9.5.1.3 Behavior Requirements**

```
1352 $instance=<CIM_IPProtocolEndpoint single object>
1353 smStopRSC ( $instance.GetObjectPath() );
1354 &smEnd;
```

1355 **6.10 CIM\_OrderedComponent**

1356 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1357 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with instances  
1358 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
1359 verb and target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and  
1360 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
1361 information in Table 10.

1362 **Table 10 – Command Verb Requirements for CIM\_OrderedComponent**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	

Command Verb	Requirement	Comments
reset	Not supported	
set	Not supported	
show	Shall	See 6.10.2.
start	Not supported	
stop	Not supported	

1363 No mappings are defined for the following verbs for the specified target: create, delete, dump, load,  
1364 reset, set, start, and stop.

### 1365 6.10.1 Ordering of Results

1366 When results are returned for multiple instances of CIM\_OrderedComponent, implementations shall  
1367 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1368 • Results for CIM\_OrderedComponent are unordered; therefore, no algorithm is defined.

### 1369 6.10.2 Show

1370 This section describes how to implement the `show` verb when applied to an instance of  
1371 CIM\_OrderedComponent. Implementations shall support the use of the `show` verb with  
1372 CIM\_OrderedComponent.

1373 The `show` command is used to display information about the CIM\_OrderedComponent instance or  
1374 instances.

#### 1375 6.10.2.1 Show Multiple Instances – CIM\_IPAssignmentSettingData Reference

1376 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
1377 to a `show` command issued against CIM\_OrderedComponent where only one reference is specified and  
1378 the reference is to an instance of CIM\_IPAssignmentSettingData.

##### 1379 6.10.2.1.1 Command Form

1380 `show <CIM_OrderedComponent multiple objects>`

##### 1381 6.10.2.1.2 CIM Requirements

1382 See CIM\_OrderedComponent in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1383 mandatory properties.

##### 1384 6.10.2.1.3 Behavior Requirements

###### 1385 6.10.2.1.3.1 Preconditions

1386 `$instance` contains the instance of CIM\_IPAssignmentSettingData which is referenced by  
1387 CIM\_OrderedComponent.

1388 There is only a single property and it is always returned.

###### 1389 6.10.2.1.3.2 Pseudo Code

```
1390 &smShowAssociationInstances ( "CIM_OrderedComponent", $instance.getObjectPath(),
1391     NULL );
1392 &smEnd;
```

**1393 6.10.2.2 Show a Single Instance – CIM\_IPAssignmentSettingData Subclass Reference**

1394 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1395 a `show` command issued against `CIM_OrderedComponent` where the reference specified is to an  
1396 instance of a subclass of `CIM_IPAssignmentSettingData`.

**1397 6.10.2.2.1 Command Form**

```
1398 show <CIM_OrderedComponent single object>
```

**1399 6.10.2.2.2 CIM Requirements**

1400 See `CIM_OrderedComponent` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1401 mandatory properties.

**1402 6.10.2.2.3 Behavior Requirements****1403 6.10.2.2.3.1 Preconditions**

1404 `$instance` contains the instance of a subclass of `CIM_IPAssignmentSettingData` which is referenced by  
1405 `CIM_OrderedComponent`.

1406 There is only a single property and it is always returned.

**1407 6.10.2.2.3.2 Pseudo Code**

```
1408 &smShowAssociationInstances ( "CIM_OrderedComponent", $instance.getObjectPath(),  
1409     NULL );  
1410 &smEnd;
```

**1411 6.10.2.3 Show a Single Instance – Both References**

1412 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1413 a `show` command issued against `CIM_OrderedComponent` where both references are specified and  
1414 therefore the desired instance is unambiguously identified.

**1415 6.10.2.3.1 Command Form**

```
1416 show <CIM_OrderedComponent single object>
```

**1417 6.10.2.3.2 CIM Requirements**

1418 See `CIM_OrderedComponent` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1419 mandatory properties.

**1420 6.10.2.3.3 Behavior Requirements****1421 6.10.2.3.3.1 Preconditions**

1422 `$instanceA` contains the instance of `CIM_IPAssignmentSettingData` that is referenced by  
1423 `CIM_OrderedComponent`.

1424 `$instanceB` contains the instance of a subclass of `CIM_IPAssignmentSettingData` which is referenced  
1425 by `CIM_OrderedComponent`.

1426 There is only a single property and it is always returned.

1427 **6.10.2.3.3.2 Pseudo Code**

```
1428 &smShowAssociationInstance ( "CIM_OrderedComponent", $instanceA.getObjectPath(),
1429     $instanceB.getObjectPath(), NULL );
1430 &smEnd;
```

1431 **6.11 CIM\_RemoteAccessAvailableToElement**

1432 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1433 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with instances  
 1434 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 1435 verb and target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and  
 1436 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1437 information in Table 11.

1438 **Table 11 – Command Verb Requirements for CIM\_RemoteAccessAvailableToElement**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.11.2.
start	Not supported	
stop	Not supported	

1439 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 1440 `reset`, `set`, `start`, and `stop`.

1441 **6.11.1 Ordering of Results**

1442 When results are returned for multiple instances of `CIM_RemoteAccessAvailableToElement`,  
 1443 implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

- 1444 • Results for `CIM_RemoteAccessAvailableToElement` are unordered; therefore, no algorithm is  
 1445 defined.

1446 **6.11.2 Show**

1447 This section describes how to implement the `show` verb when applied to an instance of  
 1448 `CIM_RemoteAccessAvailableToElement`. Implementations shall support the use of the `show` verb with  
 1449 `CIM_RemoteAccessAvailableToElement`.

1450 The `show` command is used to display information about the `CIM_RemoteAccessAvailableToElement`  
 1451 instance or instances.

**1452 6.11.2.1 Show Multiple Instances – CIM\_RemoteServiceAccessPoint Reference**

1453 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
1454 to a `show` command issued against `CIM_RemoteAccessAvailableToElement` where only one reference is  
1455 specified and the reference is to an instance of `CIM_RemoteServiceAccessPoint`.

**1456 6.11.2.1.1 Command Form**

```
1457 show <CIM_RemoteAccessAvailableToElement multiple objects>
```

**1458 6.11.2.1.2 CIM Requirements**

1459 See `CIM_RemoteAccessAvailableToElement` in the “CIM Elements” section of the [IP Interface Profile](#) for  
1460 the list of mandatory properties.

**1461 6.11.2.1.3 Behavior Requirements****1462 6.11.2.1.3.1 Preconditions**

1463 `$instance` contains the instance of `CIM_RemoteServiceAccessPoint` which is referenced by  
1464 `CIM_RemoteAccessAvailableToElement`.

1465 There is only a single property and it is always returned.

**1466 6.11.2.1.3.2 Pseudo Code**

```
1467 &smShowAssociationInstances ( "CIM_RemoteAccessAvailableToElement",  
1468     $instance.getObjectPath(), NULL );  
1469 &smEnd;
```

**1470 6.11.2.2 Show a Single Instance – CIM\_IPProtocolEndpoint Reference**

1471 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1472 a `show` command issued against `CIM_RemoteAccessAvailableToElement` where the reference specified  
1473 is to an instance of `CIM_IPProtocolEndpoint`. An instance of `CIM_IPProtocolEndpoint` is referenced by  
1474 exactly one instance of `CIM_RemoteAccessAvailableToElement`. Therefore, a single instance will be  
1475 returned.

**1476 6.11.2.2.1 Command Form**

```
1477 show <CIM_RemoteAccessAvailableToElement single object>
```

**1478 6.11.2.2.2 CIM Requirements**

1479 See `CIM_RemoteAccessAvailableToElement` in the “CIM Elements” section of the [IP Interface Profile](#) for  
1480 the list of mandatory properties.

**1481 6.11.2.2.3 Behavior Requirements****1482 6.11.2.2.3.1 Preconditions**

1483 `$instance` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
1484 `CIM_RemoteAccessAvailableToElement`.

1485 There is only a single property and it is always returned.

1486 **6.11.2.2.3.2 Pseudo Code**

```
1487 &smShowAssociationInstances ( "CIM_RemoteAccessAvailableToElement",
1488     $instance.getObjectPath(), NULL );
1489 &smEnd;
```

1490 **6.11.2.3 Show a Single Instance – Both References**

1491 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 1492 a `show` command issued against `CIM_RemoteAccessAvailableToElement` where both references are  
 1493 specified and therefore the desired instance is unambiguously identified.

1494 **6.11.2.3.1 Command Form**

```
1495 show <CIM_RemoteAccessAvailableToElement single object>
```

1496 **6.11.2.3.2 CIM Requirements**

1497 See `CIM_RemoteAccessAvailableToElement` in the “CIM Elements” section of the [IP Interface Profile](#) for  
 1498 the list of mandatory properties.

1499 **6.11.2.3.3 Behavior Requirements**

1500 **6.11.2.3.3.1 Preconditions**

1501 `$instanceA` contains the instance of `CIM_RemoteServiceAccessPoint` which is referenced by  
 1502 `CIM_RemoteAccessAvailableToElement`.

1503 `$instanceB` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
 1504 `CIM_RemoteAccessAvailableToElement`.

1505 There is only a single property and it is always returned.

1506 **6.11.2.3.3.2 Pseudo Code**

```
1507 &smShowAssociationInstance ( "CIM_RemoteAccessAvailableToElement",
1508     $instanceA.getObjectPath(), $instanceB.getObjectPath(), NULL );
1509 &smEnd;
```

1510 **6.12 CIM\_RemoteServiceAccessPoint**

1511 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1512 Table 12 lists each SM CLP verb, the required level of support for the verb in conjunction with instances  
 1513 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 1514 verb and target. Table 12 is for informational purposes only; in case of a conflict between Table 12 and  
 1515 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1516 information in Table 12.

1517 **Table 12 – Command Verb Requirements for CIM\_RemoteServiceAccessPoint**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	

Command Verb	Requirement	Comments
reset	Not supported	
set	May	See 6.12.2.
show	Shall	See 6.12.3.
start	Not supported	
stop	Not supported	

1518 No mapping is defined for the following verbs for the specified target: create, delete, dump, load,  
1519 reset, start, and stop.

### 1520 6.12.1 Ordering of Results

1521 When results are returned for multiple instances of CIM\_RemoteServiceAccessPoint, implementations  
1522 shall utilize the following algorithm to produce the natural (that is, default) ordering:

- 1523 • Results for CIM\_RemoteServiceAccessPoint are unordered; therefore, no algorithm is defined.

### 1524 6.12.2 Set

1525 This section describes how to implement the `set` verb when it is applied to an instance of  
1526 CIM\_RemoteServiceAccessPoint. Implementations may support the use of the `set` verb with  
1527 CIM\_RemoteServiceAccessPoint.

1528 The `set` verb is used to modify descriptive properties of the CIM\_RemoteServiceAccessPoint instance.

#### 1529 6.12.2.1 General Usage of Set for a Single Property

1530 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
1531 target instance. This is the most common case.

1532 The requirement for supporting modification of a property using this command form shall be equivalent to  
1533 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
1534 in the [IP Interface Profile](#).

##### 1535 6.12.2.1.1 Command Form

```
1536 set <CIM_RemoteServiceAccessPoint single instance> <propertyname>=<propertyvalue>
```

##### 1537 6.12.2.1.2 CIM Requirements

1538 See CIM\_RemoteServiceAccessPoint in the “CIM Elements” section of the [IP Interface Profile](#) for the list  
1539 of mandatory properties.

##### 1540 6.12.2.1.3 Behavior Requirements

```
1541 $instance=<CIM_RemoteServiceAccessPoint single instance>
1542 #propertyName[] = {<propertyname>};
1543 #propertyValues[] = {<propertyvalue>};
1544 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1545 &smEnd;
```



### 1546 6.12.2.2 General Usage of Set for Multiple Properties

1547 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
1548 target instance where there is not an explicit relationship between the properties. This is the most  
1549 common case.

1550 The requirement for supporting modification of a property using this command form shall be equivalent to  
1551 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
1552 in the [IP Interface Profile](#).

#### 1553 6.12.2.2.1 Command Form

```
1554 set <CIM_RemoteServiceAccessPoint single instance> <propertyname1>=<propertyvalue1>  
1555 <propertynamen>=<propertyvaluen>
```

#### 1556 6.12.2.2.2 CIM Requirements

1557 See `CIM_RemoteServiceAccessPoint` in the “CIM Elements” section of the [IP Interface Profile](#) for the list  
1558 of mandatory properties.

#### 1559 6.12.2.2.3 Behavior Requirements

```
1560 $instance=<CIM_RemoteServiceAccessPoint single instance>  
1561 #propertyNames[] = {<propertyname>};  
1562 for #i < n  
1563 {  
1564     #propertyNames[#i] = <propertyname#i>  
1565     #propertyValues[#i] = <propertyvalue#i>  
1566 }  
1567 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );  
1568 &smEnd;
```

### 1569 6.12.3 Show

1570 This section describes how to implement the `show` verb when applied to an instance of  
1571 `CIM_RemoteServiceAccessPoint`. Implementations shall support the use of the `show` verb with  
1572 `CIM_RemoteServiceAccessPoint`.

1573 The `show` verb is used to display information about the gateway.

#### 1574 6.12.3.1 Show a Single Instance

1575 This command form is for the `show` verb applied to a single instance of `CIM_RemoteServiceAccessPoint`.

##### 1576 6.12.3.1.1 Command Form

```
1577 show <CIM_RemoteServiceAccessPoint single object>
```

##### 1578 6.12.3.1.2 CIM Requirements

1579 See `CIM_RemoteServiceAccessPoint` in the “CIM Elements” section of the [IP Interface Profile](#) for the list  
1580 of mandatory properties.

##### 1581 6.12.3.1.3 Behavior Requirements

###### 1582 6.12.3.1.3.1 Preconditions

1583 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1584 **6.12.3.1.3.2 Pseudo Code**

```

1585 $instance=<CIM_RemoteServiceAccessPoint single object>
1586 #propertylist[] = NULL;
1587 if (false == #all)
1588     {
1589         #propertylist[] = { "AccessContext", "AccessInfo", "InfoFormat", "ElementName" };
1590     }
1591 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1592 &smEnd;

```

1593 **6.12.3.2 Show Multiple Instances**

1594 This command form is for the `show` verb applied to multiple instances of  
 1595 `CIM_RemoteServiceAccessPoint`. This command form corresponds to UFsT-based selection within a  
 1596 scoping system.

1597 **6.12.3.2.1 Command Form**

```

1598 show <CIM_RemoteServiceAccessPoint multiple objects>

```

1599 **6.12.3.2.2 CIM Requirements**

1600 See `CIM_RemoteServiceAccessPoint` in the "CIM Elements" section of the [IP Interface Profile](#) for the list  
 1601 of mandatory properties.

1602 **6.12.3.2.3 Behavior Requirements**1603 **6.12.3.2.3.1 Preconditions**

1604 `$containerInstance` contains the instance of `CIM_ComputerSystem` for which scoped  
 1605 `CIM_RemoteServiceAccessPoint` instances are displayed. The [IP Interface Profile](#) requires that the  
 1606 `CIM_RemoteServiceAccessPoint` instance be associated with its scoping system via an instance of the  
 1607 `CIM_HostedAccessPoint` association.

1608 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1609 **6.12.3.2.3.2 Pseudo Code**

```

1610 #propertylist[] = NULL;
1611 if (false == #all)
1612     {
1613         #propertylist[] = { "AccessContext", "AccessInfo", "InfoFormat", "ElementName" };
1614     }
1615 &smShowInstances ( "CIM_RemoteServiceAccessPoint", "CIM_HostedAccessPoint",
1616     $containerInstance.getObjectPath(), #propertylist[] );
1617 &smEnd;

```

1618 **6.13 CIM\_ServiceAffectsElement**

1619 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1620 Table 13 lists each SM CLP verb, the required level of support for the verb in conjunction with instances  
 1621 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
 1622 verb and target. Table 13 is for informational purposes only; in case of a conflict between Table 13 and  
 1623 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1624 information in Table 13.

1625

**Table 13 – Command Verb Requirements for CIM\_ServiceAffectsElement**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.13.2.
start	Not supported	
stop	Not supported	

1626 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 1627 `reset`, `set`, `start`, and `stop`.

### 1628 6.13.1 Ordering of Results

1629 When results are returned for multiple instances of `CIM_ServiceAffectsElement`, implementations shall  
 1630 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1631 • Results for `CIM_ServiceAffectsElement` are unordered; therefore, no algorithm is defined.

### 1632 6.13.2 Show

1633 This section describes how to implement the `show` verb when applied to an instance of  
 1634 `CIM_ServiceAffectsElement`. Implementations shall support the use of the `show` verb with  
 1635 `CIM_ServiceAffectsElement`.

1636 The `show` command is used to display information about the `CIM_ServiceAffectsElement` instance or  
 1637 instances.

#### 1638 6.13.2.1 Show Multiple Instances – CIM\_IPConfigurationService Reference

1639 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
 1640 to a `show` command issued against `CIM_ServiceAffectsElement` where only one reference is specified  
 1641 and the reference is to an instance of `CIM_IPConfigurationService`.

##### 1642 6.13.2.1.1 Command Form

```
1643 show <CIM_ServiceAffectsElement multiple objects>
```

##### 1644 6.13.2.1.2 CIM Requirements

1645 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
 1646 mandatory properties.

##### 1647 6.13.2.1.3 Behavior Requirements

###### 1648 6.13.2.1.3.1 Preconditions

1649 `$instance` contains the instance of `CIM_IPConfigurationService` which is referenced by  
 1650 `CIM_ServiceAffectsElement`.

1651 There is only a single property and it is always returned.

#### 1652 6.13.2.1.3.2 Pseudo Code

```
1653 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),  
1654     null );  
1655 &smEnd;
```

#### 1656 6.13.2.2 Show a Single Instance – CIM\_IPProtocolEndpoint Reference

1657 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1658 a `show` command issued against `CIM_ServiceAffectsElement` where the reference specified is to an  
1659 instance of `CIM_IPProtocolEndpoint`. An instance of `CIM_IPProtocolEndpoint` is referenced by exactly  
1660 one instance of `CIM_ServiceAffectsElement`. Therefore, a single instance will be returned.

##### 1661 6.13.2.2.1 Command Form

```
1662 show <CIM_ServiceAffectsElement single object>
```

##### 1663 6.13.2.2.2 CIM Requirements

1664 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1665 mandatory properties.

##### 1666 6.13.2.2.3 Behavior Requirements

###### 1667 6.13.2.2.3.1 Preconditions

1668 `$instance` contains the instance of `CIM_IPProtocolEndpoint` which is referenced by  
1669 `CIM_ServiceAffectsElement`.

1670 There is only a single property and it is always returned.

###### 1671 6.13.2.2.3.2 Pseudo Code

```
1672 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),  
1673     NULL );  
1674 &smEnd;
```

#### 1675 6.13.2.3 Show a Single Instance – Both References

1676 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1677 a `show` command issued against `CIM_ServiceAffectsElement` where both references are specified and  
1678 therefore the desired instance is unambiguously identified.

##### 1679 6.13.2.3.1 Command Form

```
1680 show <CIM_ServiceAffectsElement single object>
```

##### 1681 6.13.2.3.2 CIM Requirements

1682 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [IP Interface Profile](#) for the list of  
1683 mandatory properties.

1684 **6.13.2.3.3 Behavior Requirements**

1685 **6.13.2.3.3.1 Preconditions**

1686 \$instanceA contains the instance of CIM\_IPConfigurationService which is referenced by  
1687 CIM\_ServiceAffectsElement.

1688 \$instanceB contains the instance of CIM\_IPProtocolEndpoint which is referenced by  
1689 CIM\_ServiceAffectsElement.

1690 There is only a single property and it is always returned.

1691 **6.13.2.3.3.2 Pseudo Code**

```
1692 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),
1693     $instanceB.getObjectPath(), NULL );
1694 &smEnd;
```

1695 **6.14 CIM\_StaticIPAssignmentSettingData**

1696 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1697 Table 14 lists each SM CLP verb, the required level of support for the verb in conjunction with instances  
1698 of the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the  
1699 verb and target. Table 14 is for informational purposes only; in case of a conflict between Table 14 and  
1700 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
1701 information in Table 14.

1702 **Table 14 – Command Verb Requirements for CIM\_StaticIPAssignmentSettingData**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	May	See 6.14.2.
show	Shall	See 6.14.3.
start	Not supported	
stop	Not supported	

1703 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
1704 `reset`, `start`, and `stop`.

1705 **6.14.1 Ordering of Results**

1706 When results are returned for multiple instances of CIM\_StaticIPAssignmentSettingData, implementations  
1707 shall utilize the following algorithm to produce the natural (that is, default) ordering:

- 1708 • Results for CIM\_StaticIPAssignmentSettingData are unordered; therefore, no algorithm is  
1709 defined.

1710 **6.14.2 Set**

1711 This section describes how to implement the `set` verb when it is applied to an instance of  
 1712 `CIM_StaticIPAssignmentSettingData`. Implementations may support the use of the `set` verb with  
 1713 `CIM_StaticIPAssignmentSettingData`.

1714 The `set` verb is used to modify descriptive properties of an instance of  
 1715 `CIM_StaticIPAssignmentSettingData`.

1716 **6.14.2.1 General Usage of Set for a Single Property**

1717 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
 1718 target instance. This is the most common case.

1719 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1720 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
 1721 in the [IP Interface Profile](#).

1722 **6.14.2.1.1 Command Form**

```
1723 set <CIM_StaticIPAssignmentSettingData single instance> <propertyname>=<propertyvalue>
```

1724 **6.14.2.1.2 CIM Requirements**

1725 See `CIM_StaticIPAssignmentSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the  
 1726 list of modifiable properties.

1727 **6.14.2.1.3 Behavior Requirements**

```
1728 $instance=<CIM_StaticIPAssignmentSettingData single instance>
1729 #propertyNames[] = {<propertyname>};
1730 #propertyValues[] = {<propertyvalue>};
1731 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
1732 &smEnd;
```

1733 **6.14.2.2 General Usage of Set for Multiple Properties**

1734 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
 1735 target instance where there is not an explicit relationship between the properties. This is the most  
 1736 common case.

1737 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1738 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
 1739 in the [IP Interface Profile](#).

1740 **6.14.2.2.1 Command Form**

```
1741 set <CIM_StaticIPAssignmentSettingData single instance>
1742 <propertyname1>=<propertyvalue1> <propertynamen>=<propertyvaluen>
```

1743 **6.14.2.2.2 CIM Requirements**

1744 See `CIM_StaticIPAssignmentSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the  
 1745 list of mandatory properties.

### 1746 6.14.2.2.3 Behavior Requirements

```

1747 $instance=<CIM_StaticIPAssignmentSettingData single instance>
1748 #propertyName[] = {<propertyname>};
1749 for #i < n
1750 {
1751     #propertyName[#i] = <propertyname#i>
1752     #propertyValues[#i] = <propertyvalue#i>
1753 }
1754 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1755 &smEnd;

```

### 1756 6.14.3 Show

1757 This section describes how to implement the `show` verb when applied to an instance of  
 1758 `CIM_StaticIPAssignmentSettingData`. Implementations shall support the use of the `show` verb with  
 1759 `CIM_StaticIPAssignmentSettingData`.

1760 The `show` verb is used to display information about the `CIM_StaticIPAssignmentSettingData` instance.

#### 1761 6.14.3.1 Show a Single Instance

1762 This command form is for the `show` verb applied to a single instance of  
 1763 `CIM_StaticIPAssignmentSettingData`.

##### 1764 6.14.3.1.1 Command Form

```

1765 show <CIM_StaticIPAssignmentSettingData single object>

```

##### 1766 6.14.3.1.2 CIM Requirements

1767 See `CIM_StaticIPAssignmentSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the  
 1768 list of mandatory properties.

##### 1769 6.14.3.1.3 Behavior Requirements

###### 1770 6.14.3.1.3.1 Preconditions

1771 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

###### 1772 6.14.3.1.3.2 Pseudo Code

```

1773 $instance=<CIM_StaticIPAssignmentSettingData single object>
1774 #propertylist[] = NULL;
1775 if (false == #all)
1776 {
1777     #propertylist[] = {“ElementName”, “IPv4Address”, “SubnetMask”,
1778         “GatewayIPv4Address”, “AddressOrigin” }
1779 }
1780 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1781 &smEnd;

```

**1782 6.14.3.2 Show Multiple Instances**

1783 This command form is for the `show` verb applied to multiple instances of  
1784 `CIM_StaticIPAssignmentSettingData`. This command form corresponds to UFsT-based selection within a  
1785 scoping system.

**1786 6.14.3.2.1 Command Form**

```
1787 show <CIM_StaticIPAssignmentSettingData multiple objects>
```

**1788 6.14.3.2.2 CIM Requirements**

1789 See `CIM_StaticIPAssignmentSettingData` in the “CIM Elements” section of the [IP Interface Profile](#) for the  
1790 list of mandatory properties.

**1791 6.14.3.2.3 Behavior Requirements****1792 6.14.3.2.3.1 Preconditions**

1793 `$containerInstance` contains the instance of `CIM_IPAssignmentSettingData` for which scoped  
1794 `CIM_StaticIPAssignmentSettingData` instances are displayed. The [IP Interface Profile](#) requires that the  
1795 `CIM_StaticIPAssignmentSettingData` instance be associated with an instance of  
1796 `CIM_IPAssignmentSettingData` via an instance of the `CIM_OrderedComponent` association.

1797 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

**1798 6.14.3.2.3.2 Pseudo Code**

```
1799 #propertylist[] = NULL;  
1800 if (false == #all)  
1801 {  
1802     #propertylist[] = {“ElementName”, “IPv4Address”, “SubnetMask”,  
1803         “GatewayIPv4Address”, “AddressOrigin” }  
1804 }  
1805 &smShowInstances ( “CIM_StaticIPAssignmentSettingData”, “CIM_OrderedComponent”,  
1806     $containerInstance.getObjectPath(), #propertylist[] );  
1807 &smEnd;
```

1808



**ANNEX A**  
(informative)

**Change Log**

1809  
1810  
1811  
1812  
1813

Version	Date	Author	Description
1.0.0	2009-07-14		DMTF Standard Release

1814